浙江大學
ZHEJIANG UNIVERSITY

Computer Architecture Experiment

Topic 5. Pipelined CPU supporting 31 MIPS Instructions

浙江大学计算机学院
陈文智
chenwz@zju.edu.cn

实验操作流程

- 阅读实验文档，理解31条MIPS指令实现方式
- 以前一次实验为基础，支持31条MIPS指令，使处理器功能更加丰富
- 对处理器进行仿真，检验处理器的仿真结果是否符合要求。
- 综合工程并下载至开发板，在单步执行的过程中检查调试屏幕的输出，检验处理器的执行过程是否正确。

实验验收标准

- 仿真执行过程中，处理器的行为和内部控制信号均符合要求。
- 下载至开发板后的单步执行过程中，寄存器的变化过程和最终执行结果与测试程序相吻合。

Outline

- Experiment Purpose
- Experiment Task
- Basic Principle
- Operating Procedures
- Precaution
- Checkpoints

Experiment Purpose

- Understand 31 MIPS instructions.
- Master the design methods of pipelined CPU executing 31 MIPS instructions.
- master methods of program verification of Pipelined CPU executing 31 MIPS instructions

Experiment Task

- **Design of Pipelined CPU executing 31 MIPS instructions.**
  - Design datapath
  - Design CPU Controller
- **Verify the Pipelined CPU with program and observe the execution of program**

Pipelined CPU supporting execution of 31 MIPS instructions

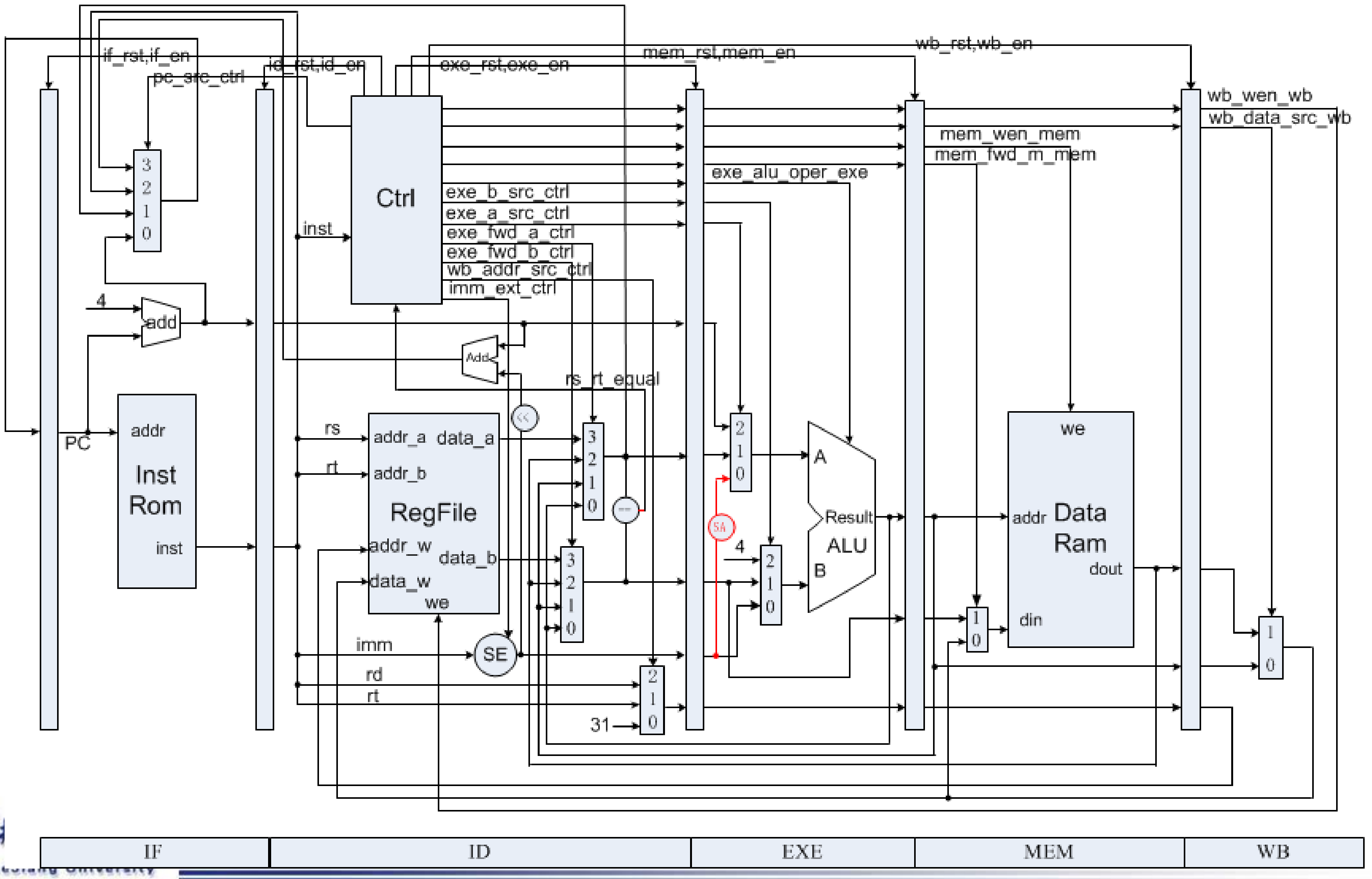
MIPS Instructions								
Bit #	**31..26**	**25..21**	**20..16**	**15..11**	**10..6**	**5..0**	**Operations**	
R-type	op	rs	rt	rd	sa	func		
add		rs	rt	rd	00000	100000	rd = rs + rt; with overflow	PC+=4
addu		rs	rt	rd	00000	100001	rd = rs + rt; without overflow	PC+=4
sub		rs	rt	rd	00000	100010	rd = rs - rt; with overflow	PC+=4
subu		rs	rt	rd	00000	100011	rd = rs - rt; without overflow	PC+=4
and		rs	rt	rd	00000	100100	rd = rs & rt;	PC+=4
or		rs	rt	rd	00000	100101	rd = rs | rt;	PC+=4
xor		rs	rt	rd	00000	100110	rd = rs ^ rt;	PC+=4
nor		rs	rt	rd	00000	100111	rd = ~(rs | rt);	PC+=4
slt	000000	rs	rt	rd	00000	101010	if(rs < rt)rd = 1; else rd = 0; <(signed)	PC+=4
sltu		rs	rt	rd	00000	101011	if(rs < rt)rd = 1; else rd = 0; <(unsigned)	PC+=4
sll		00000	rt	rd	sa	000000	rd = rt << sa;	PC+=4
srl		00000	rt	rd	sa	000010	rd = rt >> sa (logical);	PC+=4
sra		00000	rt	rd	sa	000011	rd = rt >> sa (arithmetic);	PC+=4
sllv		rs	rt	rd	00000	000100	rd = rt << rs;	PC+=4
srlv		rs	rt	rd	00000	000110	rd = rt >> rs (logical);	PC+=4
srav		rs	rt	rd	00000	000111	rd = rt >> rs(arithmetic);	PC+=4
jr		rs	00000	00000	00000	001000		PC=rs

Pipelined CPU supporting execution of 31 MIPS instructions

MIPS Instructions

Bit #	31..26	25..21	20..16	15..11	10..6	5..0	Operations	
I-type	op	rs	rt	immediate				
addi	001000	rs	rt	imm			rt = rs + (sign_extend)imm; with overflow	PC+=4
addiu	001001	rs	rt	imm			rt = rs + (sign_extend)imm;without overflow	PC+=4
andi	001100	rs	rt	imm			rt = rs & (zero_extend)imm;	PC+=4
ori	001101	rs	rt	imm			rt = rs | (zero_extend)imm;	PC+=4
xori	001110	rs	rt	imm			rt = rs ^ (zero_extend)imm;	PC+=4
lui	001111	00000	rt	imm			rt = imm * 65536;	PC+=4
lw	100011	rs	rt	imm			rt = memory[rs + (sign_extend)imm];	PC+=4
sw	101011	rs	rt	imm			memory[rs + (sign_extend)imm] <-- rt;	PC+=4
beq	000100	rs	rt	imm			if (rs == rt) PC+=4 + (sign_extend)imm <<2;	PC+=4
bne	000101	rs	rt	imm			if (rs != rt) PC+=4 + (sign_extend)imm <<2;	PC+=4
slti	001010	rs	rt	imm			if (rs < (sign_extend)imm) rt =1 else rt = 0; less than signed	PC+=4
sltiu	001011	rs	rt	imm			if (rs < (zero_extend)imm) rt =1 else rt = 0; less than unsigned	PC+=4
J-type	op	address						
j	000010	address					PC = (PC+4)[31..28],address<<2	
jal	000011	address					PC = (PC+4)[31..28],address<<2 ; $31 = PC+4	

Datapath of CPU supporting 31 MIPS Instr.



signed and unsigned type

- **reg and wire are unsigned type in Default**

- **Signed Type:**
  - reg signed
  - wire signed

- **Type Conversion Function**
  - $signed()
  - $unsiged()

ALU

```
case (oper)
        EXE_ALU_SLT: begin
                if (sign)
                        result = $signed(a) < $signed(b);
                else
                        result = ……
        EXE_ALU_LUI: begin
                result = ……
        end
        ……
        EXE_ALU_SR: begin
                if (sign)
                        ……
                else
                        ……
endcase
```

Instr. Mem.(1)

```
0 : 3c010000;   % (00) main:    lui $1, 0 # address of data[0] %
1 : 34240050;   % (04)          ori $4, $1, 80 # address of data[0] %
2 : 0c00001b;   % (08) call:    jal sum # call function %
3 : 20050004;   % (0c) dslot1:  addi $5, $0, 4 # counter,DELYED SLOT(DS) %
4 : ac820000;   % (10) return:  sw $2, 0($4) # store result %
5 : 8c890000;   % (14)          lw $9, 0($4) # check sw %
6 : 01244022;   % (18)          sub $8, $9, $4 # sub: $8 <- $9 - $4 %
7 : 20050003;   % (1c)          addi $5, $0, 3 # counter %
8 : 20a5ffff;   % (20) loop2:   addi $5, $5, -1 # counter - 1 %
9 : 34a8ffff;   % (24)          ori $8, $5, 0xffff # zero-extend: 0000ffff %
A : 39085555;   % (28)          xori $8, $8, 0x5555 # zero-extend: 0000aaaa %
B : 2009ffff;   % (2c)          addi $9, $0, -1 # sign-extend: ffffffff %
```

Instr. Mem.(2)

```
C : 312affff;    % (30)           andi $10, $9, 0xffff # zero-extend: 0000ffff %
D : 01493025;    % (34)           or $6, $10, $9 # or: ffffffff %
E : 01494026;    % (38)            xor $8, $10, $9 # xor: ffff0000 %
F : 01463824;    % (3c)           and $7, $10, $6 # and: 0000ffff %
10 : 10a00003;  % (40)           beq $5, $0, shift # if $5 = 0, goto shift %
11 : 00000000;  % (44) dslot2:   nop # DS %
12 : 08000008;  % (48)           j loop2 # jump loop2 %
13 : 00000000;  % (4c) dslot3:   nop # DS %
14 : 2005ffff;    % (50) shift:    addi $5, $0, -1 # $5 = ffffffff %
15 : 000543c0;  % (54)           sll $8, $5, 15 # <<15 = ffff8000 %
16 : 00084400;  % (58)           sll $8, $8, 16 # <<16 = 80000000 %
17 : 00084403;  % (5c)           sra $8, $8, 16 # >>16 = ffff8000 (arith) %
```

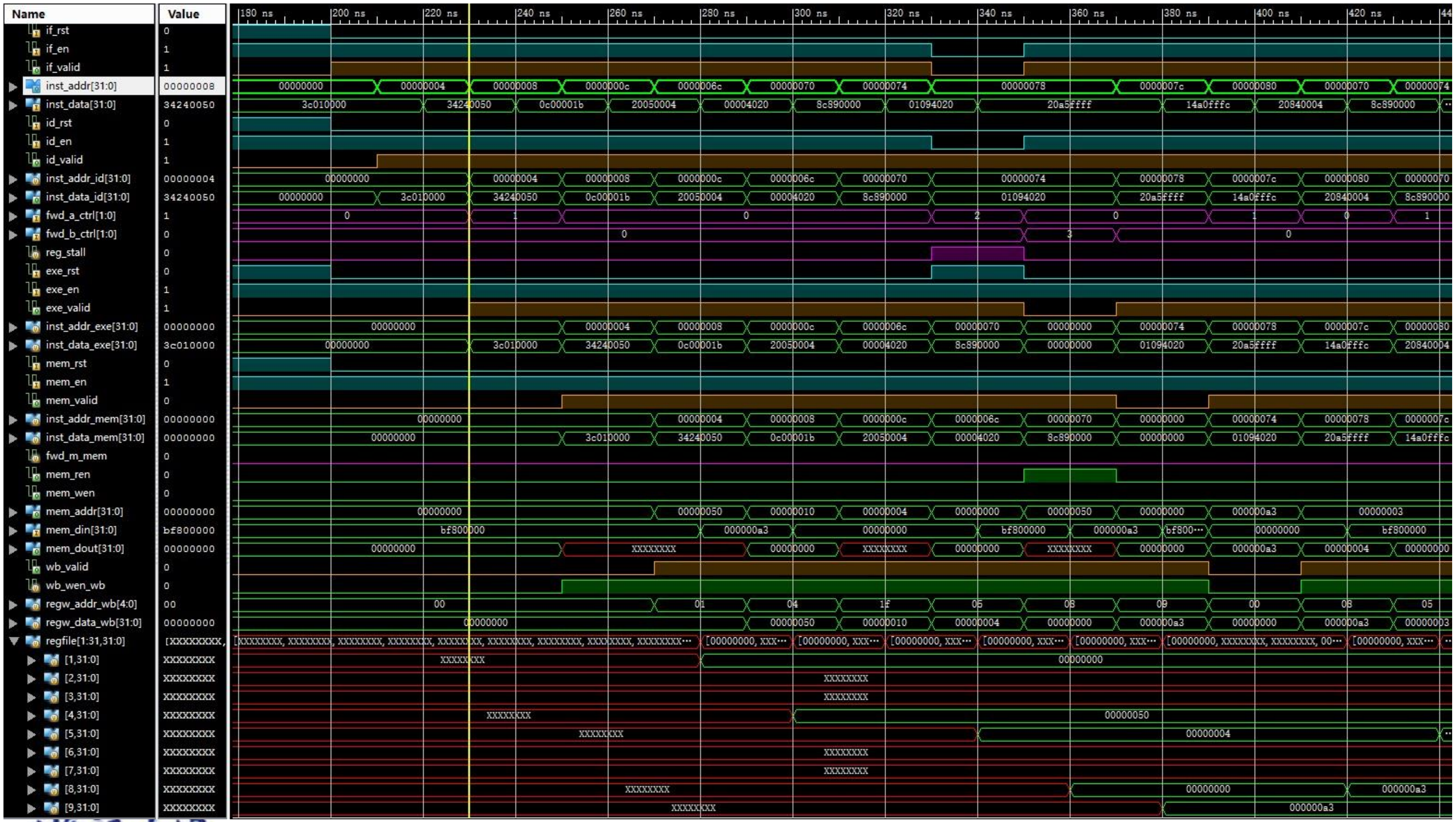
Instr. Mem.(3)

```
18 : 000843c2;  % (60)           srl $8, $8, 15 # >>15 = 0001ffff (logic) %
19 : 08000019;  % (64) finish:   j finish # dead loop %
1A : 00000000;  % (68) dslot4:   nop # DS %
1B : 00004020;  % (6c) sum:      add $8, $0, $0 # sum function entry %
1C : 8c890000;  % (70) loop:     lw $9, 0($4) # load data %
1D : 01094020;  % (74)           add $8, $8, $9 # sum, PIPELINE STALLS %
1E : 20a5ffff;  % (78)           addi $5, $5, -1 # counter - 1 %
1F : 14a0fffc;  % (7c)           bne $5, $0, loop # finish? %
20 : 20840004;  % (80) dslot5:   addi $4, $4, 4 # address + 4, DS %
21 : 03e00008;  % (84)           jr $ra # return %
22 : 00081000;  % (88) dslot6:   sll $2, $8, 0 # move result to $v0, DS %
```

Data Mem.

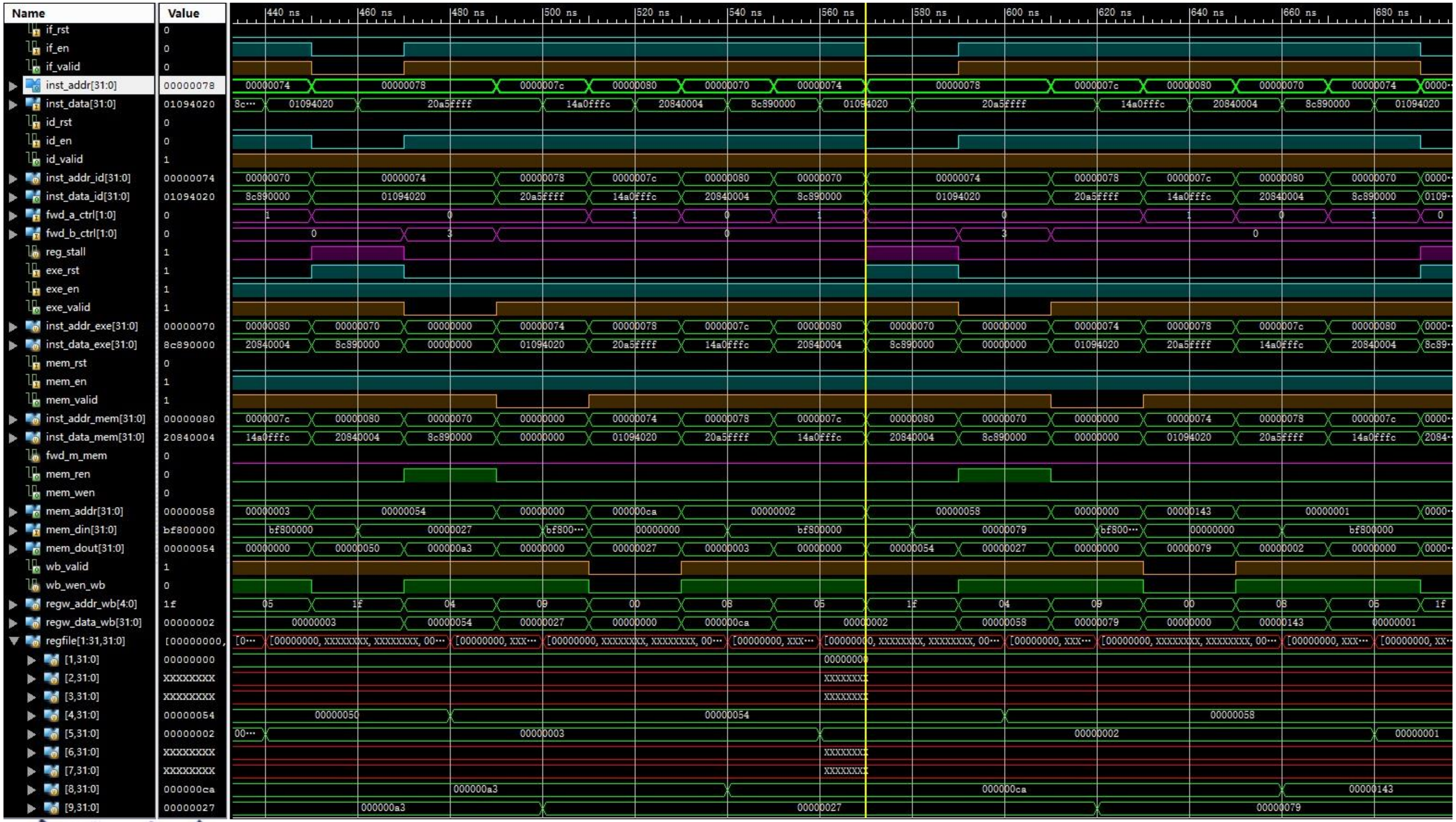
```
0 :  BF800000;          % 1 01111111 00..0 fp -1 %
14 : 000000A3;          % (50) data[0] %
15 : 00000027;          % (54) data[1] %
16 : 00000079;          % (58) data[2] %
17 : 00000115;          % (5C) data[3] %
```

Simulation (1)



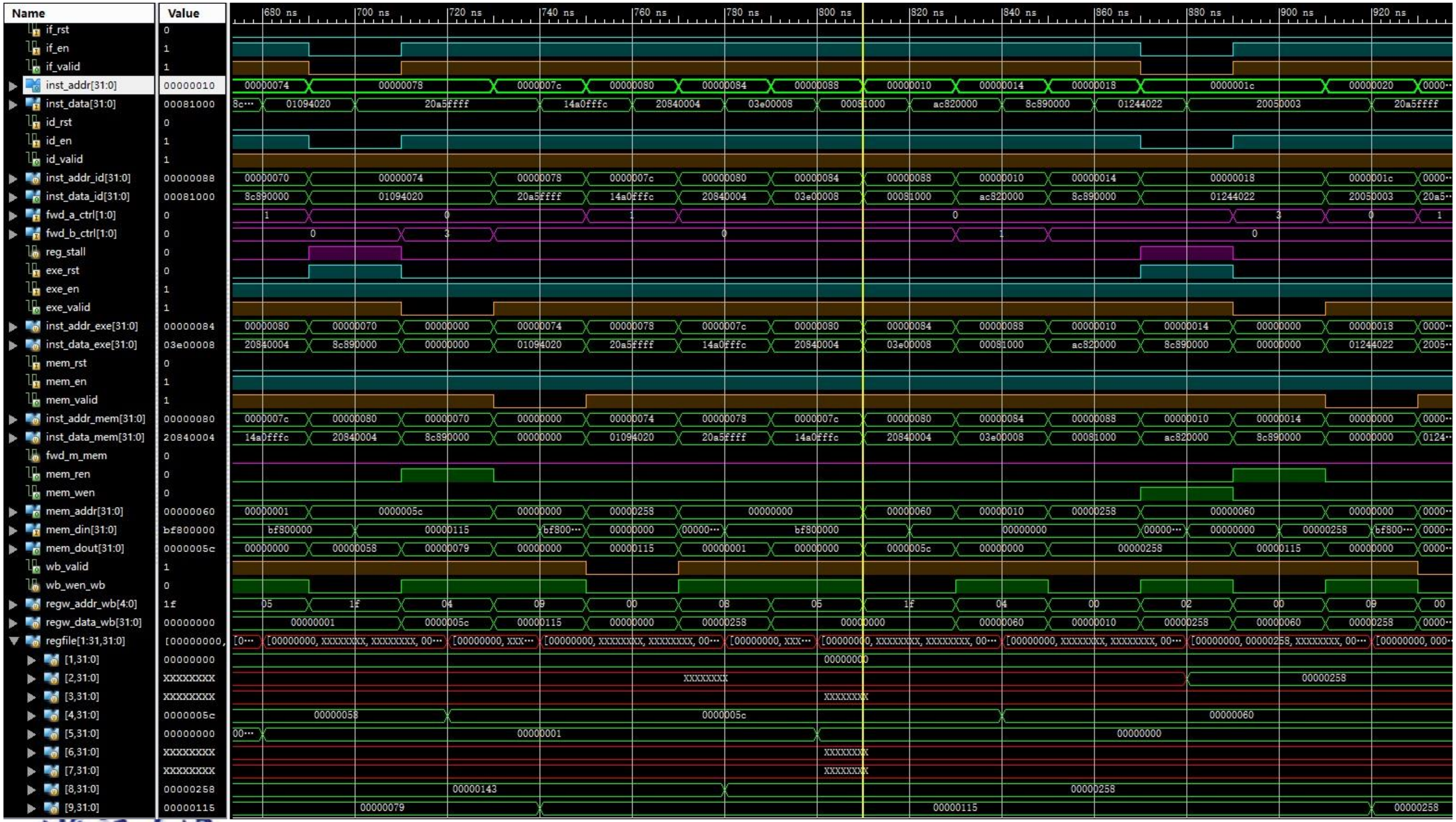
Name	Value
if_rst	0
if_en	1
if_valid	1
inst_addr[31:0]	00000008
inst_data[31:0]	34240050
id_rst	0
id_en	1
id_valid	1
inst_addr_id[31:0]	00000004
inst_data_id[31:0]	34240050
fwd_a_ctrl[1:0]	1
fwd_b_ctrl[1:0]	0
reg_stall	0
exe_rst	0
exe_en	1
exe_valid	1
inst_addr_exe[31:0]	00000000
inst_data_exe[31:0]	3c010000
mem_rst	0
mem_en	1
mem_valid	0
inst_addr_mem[31:0]	00000000
inst_data_mem[31:0]	00000000
fwd_m_mem	0
mem_ren	0
mem_wen	0
mem_addr[31:0]	00000000
mem_din[31:0]	bf800000
mem_dout[31:0]	00000000
wb_valid	0
wb_wen_wb	0
regw_addr_wb[4:0]	00
regw_data_wb[31:0]	00000000
regfile[1:31,31:0]	[xxxxxxxx,
[1,31:0]	xxxxxxxx
[2,31:0]	xxxxxxxx
[3,31:0]	xxxxxxxx
[4,31:0]	xxxxxxxx
[5,31:0]	xxxxxxxx
[6,31:0]	xxxxxxxx
[7,31:0]	xxxxxxxx
[8,31:0]	xxxxxxxx
[9,31:0]	xxxxxxxx

Simulation (2)



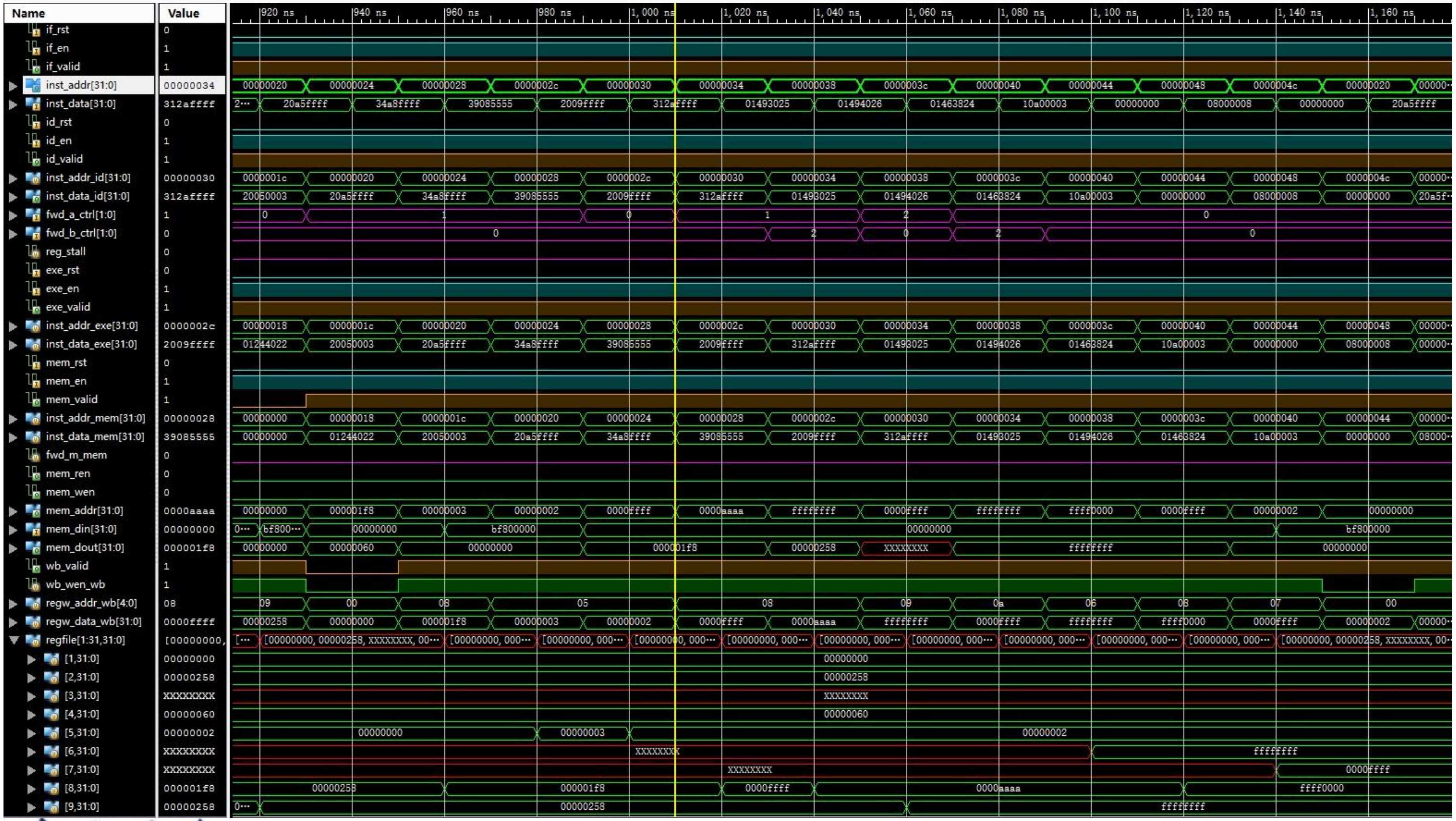
Name	Value
if_rst	0
if_en	0
if_valid	0
inst_addr[31:0]	00000078
inst_data[31:0]	01094020
id_rst	0
id_en	0
id_valid	1
inst_addr_id[31:0]	00000074
inst_data_id[31:0]	01094020
fwd_a_ctrl[1:0]	0
fwd_b_ctrl[1:0]	0
reg_stall	1
exe_rst	1
exe_en	1
exe_valid	1
inst_addr_exe[31:0]	00000070
inst_data_exe[31:0]	8c890000
mem_rst	0
mem_en	1
mem_valid	1
inst_addr_mem[31:0]	00000080
inst_data_mem[31:0]	20840004
fwd_m_mem	0
mem_ren	0
mem_wen	0
mem_addr[31:0]	00000058
mem_din[31:0]	bf800000
mem_dout[31:0]	00000054
wb_valid	1
wb_wen_wb	0
regw_addr_wb[4:0]	1f
regw_data_wb[31:0]	00000002
regfile[1:31,31:0]	[00000000,
[1,31:0]	00000000
[2,31:0]	xxxxxxxx
[3,31:0]	xxxxxxxx
[4,31:0]	00000054
[5,31:0]	00000002
[6,31:0]	xxxxxxxx
[7,31:0]	xxxxxxxx
[8,31:0]	000000ca
[9,31:0]	00000027

Simulation (3)

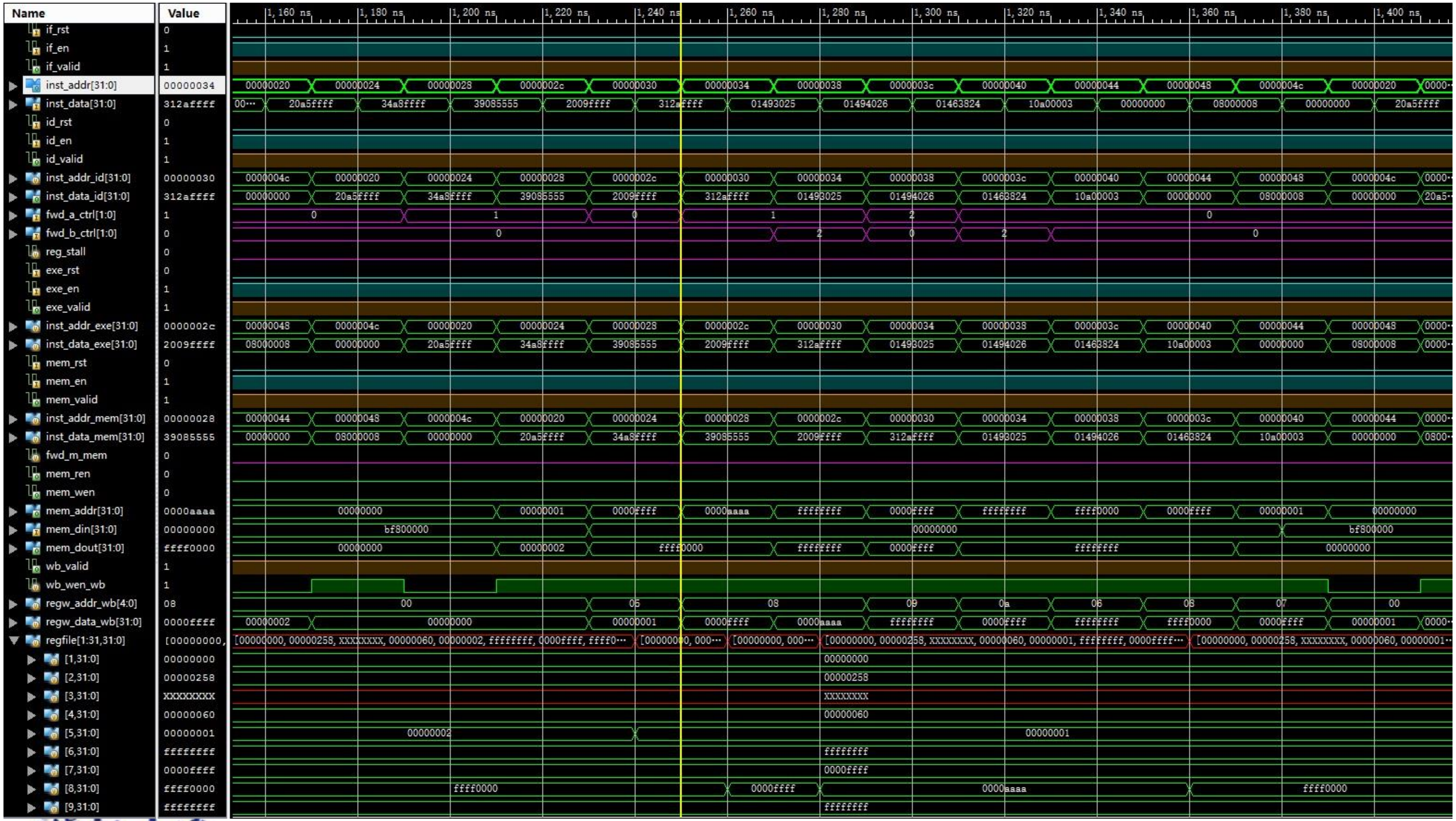


Name	Value
if_rst	0
if_en	1
if_valid	1
inst_addr[31:0]	00000010
inst_data[31:0]	00081000
id_rst	0
id_en	1
id_valid	1
inst_addr_id[31:0]	00000088
inst_data_id[31:0]	00081000
fwd_a_ctrl[1:0]	0
fwd_b_ctrl[1:0]	0
reg_stall	0
exe_rst	0
exe_en	1
exe_valid	1
inst_addr_exe[31:0]	00000084
inst_data_exe[31:0]	03e00008
mem_rst	0
mem_en	1
mem_valid	1
inst_addr_mem[31:0]	00000080
inst_data_mem[31:0]	20840004
fwd_m_mem	0
mem_ren	0
mem_wen	0
mem_addr[31:0]	00000060
mem_din[31:0]	bf800000
mem_dout[31:0]	0000005c
wb_valid	1
wb_wen_wb	0
regw_addr_wb[4:0]	1f
regw_data_wb[31:0]	00000000
regfile[1:31,31:0]	[00000000,
[1,31:0]	00000000
[2,31:0]	xxxxxxxx
[3,31:0]	xxxxxxxx
[4,31:0]	0000005c
[5,31:0]	00000000
[6,31:0]	xxxxxxxx
[7,31:0]	xxxxxxxx
[8,31:0]	00000258
[9,31:0]	00000115

Simulation (4)

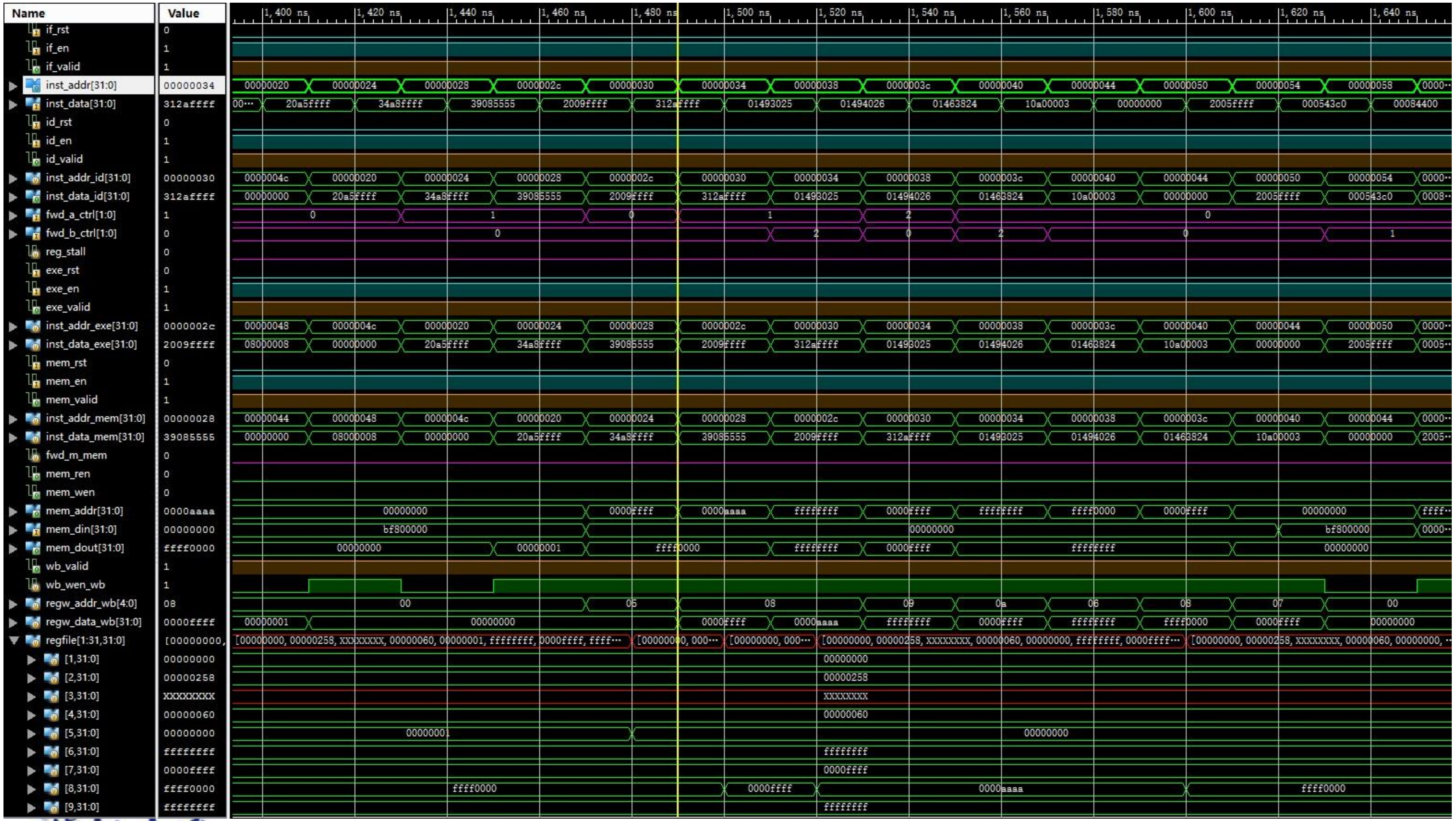


Simulation (5)



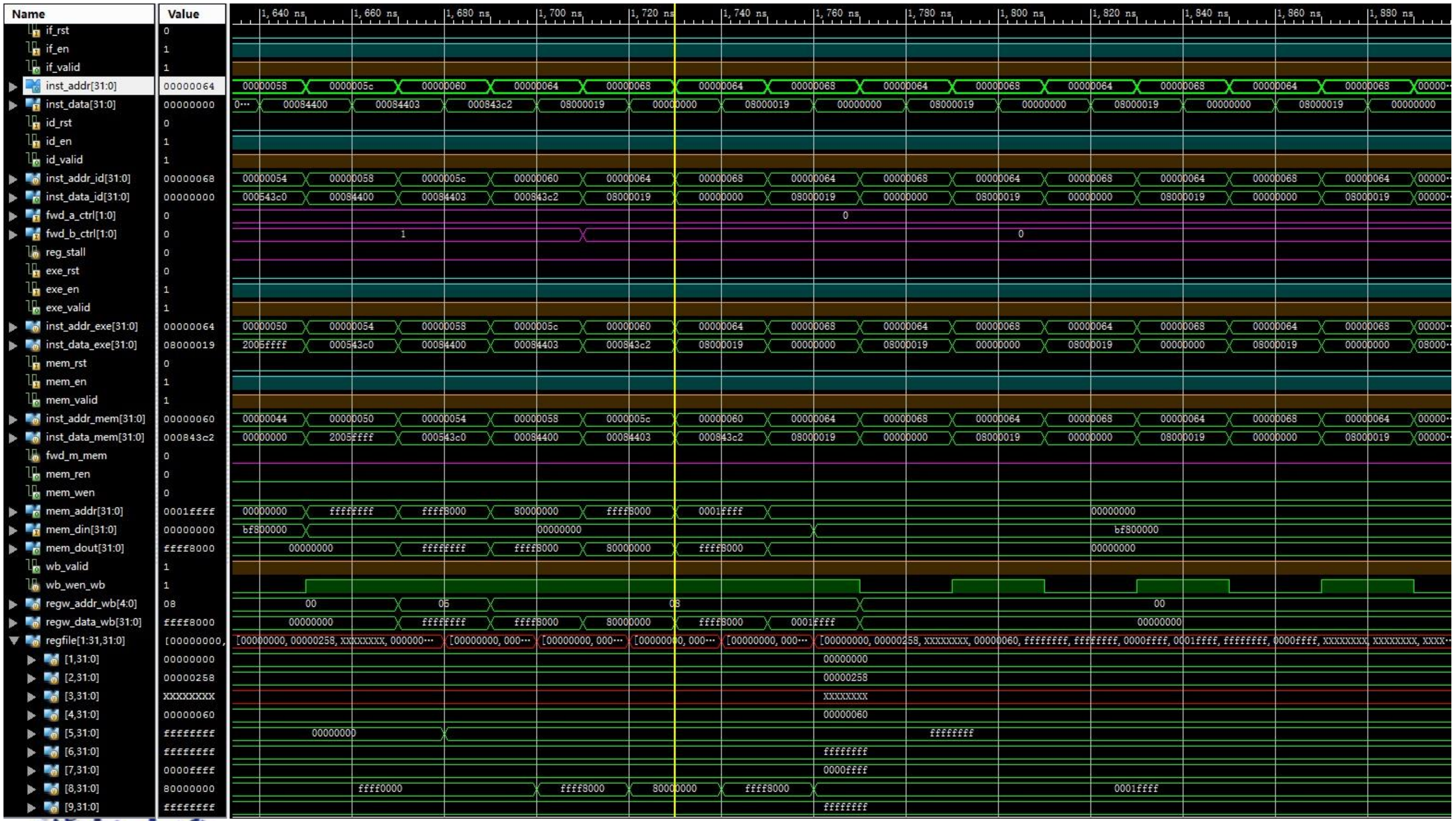
Name	Value
if_rst	0
if_en	1
if_valid	1
inst_addr[31:0]	00000034
inst_data[31:0]	312affff
id_rst	0
id_en	1
id_valid	1
inst_addr_id[31:0]	00000030
inst_data_id[31:0]	312affff
fwd_a_ctrl[1:0]	1
fwd_b_ctrl[1:0]	0
reg_stall	0
exe_rst	0
exe_en	1
exe_valid	1
inst_addr_exe[31:0]	0000002c
inst_data_exe[31:0]	2009ffff
mem_rst	0
mem_en	1
mem_valid	1
inst_addr_mem[31:0]	00000028
inst_data_mem[31:0]	39085555
fwd_m_mem	0
mem_ren	0
mem_wen	0
mem_addr[31:0]	0000aaaa
mem_din[31:0]	00000000
mem_dout[31:0]	ffff0000
wb_valid	1
wb_wen_wb	1
regw_addr_wb[4:0]	08
regw_data_wb[31:0]	0000ffff
regfile[1:31,31:0]	[00000000,
[1,31:0]	00000000
[2,31:0]	00000258
[3,31:0]	xxxxxxxx
[4,31:0]	00000060
[5,31:0]	00000001
[6,31:0]	ffffffff
[7,31:0]	0000ffff
[8,31:0]	ffff0000
[9,31:0]	ffffffff

Simulation (6)



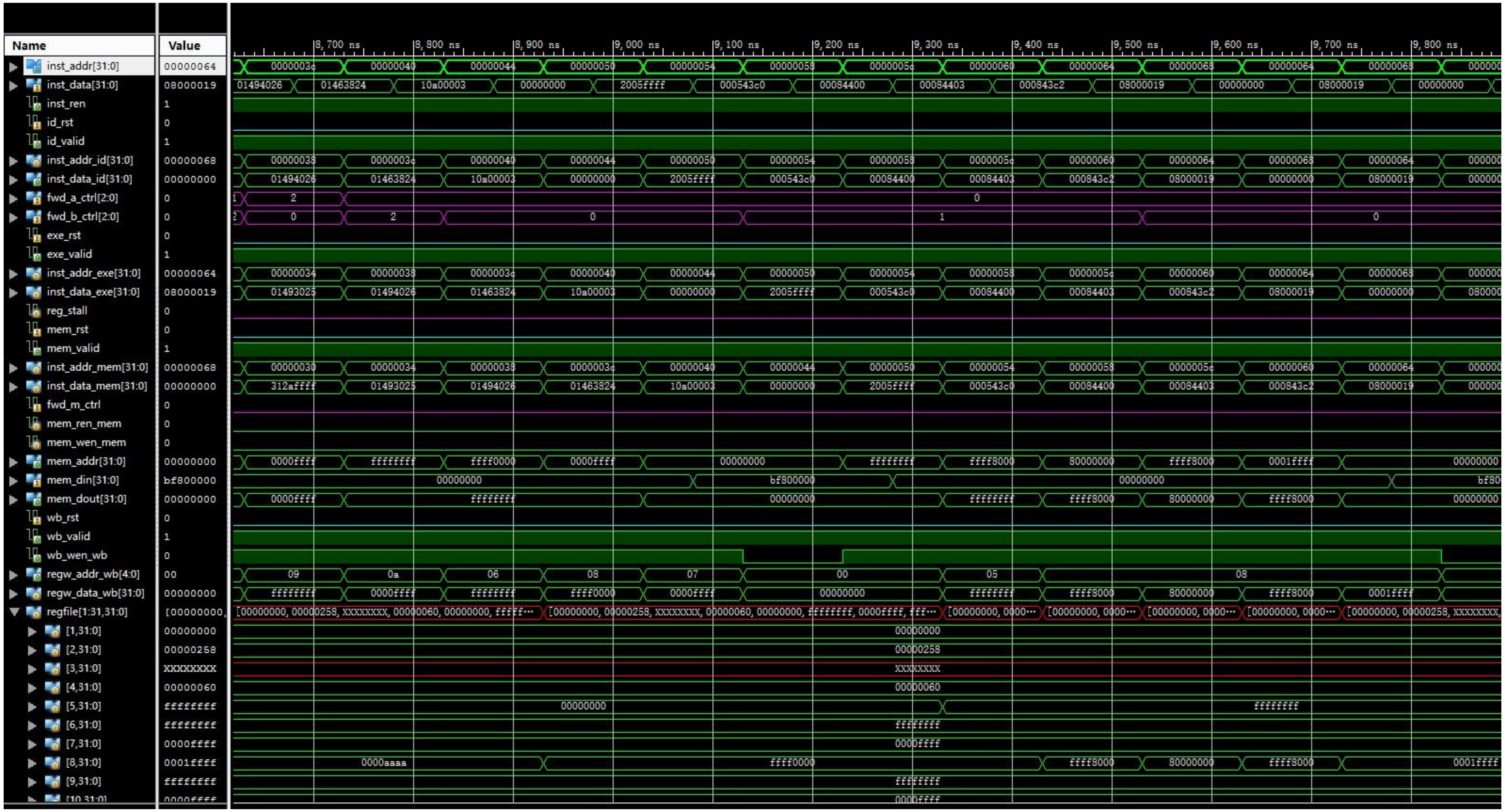
Name	Value
if_rst	0
if_en	1
if_valid	1
inst_addr[31:0]	00000034
inst_data[31:0]	312affff
id_rst	0
id_en	1
id_valid	1
inst_addr_id[31:0]	00000030
inst_data_id[31:0]	312affff
fwd_a_ctrl[1:0]	1
fwd_b_ctrl[1:0]	0
reg_stall	0
exe_rst	0
exe_en	1
exe_valid	1
inst_addr_exe[31:0]	0000002c
inst_data_exe[31:0]	2009ffff
mem_rst	0
mem_en	1
mem_valid	1
inst_addr_mem[31:0]	00000028
inst_data_mem[31:0]	39085555
fwd_m_mem	0
mem_ren	0
mem_wen	0
mem_addr[31:0]	0000aaaa
mem_din[31:0]	00000000
mem_dout[31:0]	ffff0000
wb_valid	1
wb_wen_wb	1
regw_addr_wb[4:0]	08
regw_data_wb[31:0]	0000ffff
regfile[1:31,31:0]	[00000000,
[1,31:0]	00000000
[2,31:0]	00000258
[3,31:0]	xxxxxxxx
[4,31:0]	00000060
[5,31:0]	00000000
[6,31:0]	ffffffff
[7,31:0]	0000ffff
[8,31:0]	ffff0000
[9,31:0]	ffffffff

Simulation (7)



Name	Value
if_rst	0
if_en	1
if_valid	1
inst_addr[31:0]	00000064
inst_data[31:0]	00000000
id_rst	0
id_en	1
id_valid	1
inst_addr_id[31:0]	00000068
inst_data_id[31:0]	00000000
fwd_a_ctrl[1:0]	0
fwd_b_ctrl[1:0]	0
reg_stall	0
exe_rst	0
exe_en	1
exe_valid	1
inst_addr_exe[31:0]	00000064
inst_data_exe[31:0]	08000019
mem_rst	0
mem_en	1
mem_valid	1
inst_addr_mem[31:0]	00000060
inst_data_mem[31:0]	000843c2
fwd_m_mem	0
mem_ren	0
mem_wen	0
mem_addr[31:0]	0001ffff
mem_din[31:0]	00000000
mem_dout[31:0]	ffff8000
wb_valid	1
wb_wen_wb	1
regw_addr_wb[4:0]	08
regw_data_wb[31:0]	ffff8000
regfile[1:31,31:0]	[00000000,
[1,31:0]	00000000
[2,31:0]	00000258
[3,31:0]	xxxxxxxx
[4,31:0]	00000060
[5,31:0]	ffffffff
[6,31:0]	ffffffff
[7,31:0]	0000ffff
[8,31:0]	80000000
[9,31:0]	ffffffff

Simulation (8)



```
Name                 Value
inst_addr[31:0]      00000064
inst_data[31:0]      08000019
inst_ren             1
id_rst               0
id_valid             1
inst_addr_id[31:0]   00000068
inst_data_id[31:0]   00000000
fwd_a_ctrl[2:0]      0
fwd_b_ctrl[2:0]      0
exe_rst              0
exe_valid            1
inst_addr_exe[31:0]  00000064
inst_data_exe[31:0]  08000019
reg_stall            0
mem_rst              0
mem_valid            1
inst_addr_mem[31:0]  00000068
inst_data_mem[31:0]  00000000
fwd_m_ctrl           0
mem_ren_mem          0
mem_wen_mem          0
mem_addr[31:0]       00000000
mem_din[31:0]        bf800000
mem_dout[31:0]       00000000
wb_rst               0
wb_valid             1
wb_wen_wb            0
regw_addr_wb[4:0]    00
regw_data_wb[31:0]   00000000
regfile[1:31,31:0]   [00000000,
  [1,31:0]           00000000
  [2,31:0]           00000258
  [3,31:0]           xxxxxxxx
  [4,31:0]           00000060
  [5,31:0]           ffffffff
  [6,31:0]           ffffffff
  [7,31:0]           0000ffff
  [8,31:0]           0001ffff
  [9,31:0]           ffffffff
  [10,31:0]          0000ffff
```

Checkpoints

- **CP 1:**

  Waveform Simulation of the Pipelined CPU with the verification program

- **CP 2:**

  FPGA Implementation of the Pipelined CPU with the verification program

Thanks!